Inverse Problems, Design and Optimization Symposium
Rio de Janeiro, Brazil, 2004

IMPROVING THE DESIGN OF CLUSTERED NEURAL FUZZY MODELS FOR OPTIMIZATION

Frederico G. Guimarães
Department of Electrical Engineering
Federal University of Minas Gerais, UFMG
Minas Gerais, MG, Brazil
fgg@ufmg.br

Jaime A. Ramírez
Department of ElectricalEngineering
Federal University of Minas Gerais, UFMG
Minas Gerais, MG, Brazil
jramirez@cpdee.ufmg.br

ABSTRACT

This paper discusses the application of hierarchical clustering techniques in the design of clustered neural fuzzy approximations for optimization problems. Different clustering techniques are investigated for addressing the function approximation problem from data samples. The hierarchical clustering process has the advantage of providing a rational manner to determine the adequate number of clusters. We present a methodology for the iterative design of the neural fuzzy network model using the clustering scheme. Experiments with some analytical functions are performed to confirm the methodology discussed. Finally, we investigate the design of a superconducting magnetic energy storage device with two solenoids and three design parameters. The results indicate that the employment of hierarchical clustering techniques for generating neural fuzzy approximations is a valuable technique to solve and to reduce the computational cost of practical optimization problems.

NOMENCLATURE

$\mathbf{c}_i$	The centroid of the data in the *i*-th cluster
C_i	Represents the *i*-th cluster
$d(C_p, C_q)$	The distance between clusters C_p and C_q
$f(\cdot): \Re^n \mapsto \Re$	The objective function
K	The total number of clusters
n	The number of optimization parameters or variables
p_{ij}	Linear parameters of the Fuzzy Inference System
$\Re$	The Euclidian real space
s_{ij}	The standard deviation of the *j*-th gaussian membership function for the *i*-th variable
T	The total number of data pair samples in Γ
$\mathbf{x}$	The *n*-dimensional vector of optimization parameters
$\mathbf{x}_t$	The *t*-th input vector in Γ
x_i	The *i*-th variable in $\mathbf{x}$
x_{ij}	The *j*-th variable in $\mathbf{x}_i$
$y_t = f(\mathbf{x}_t)$	The evaluation of the *t*-th input vector in Γ
Γ	Represents the data training set

INTRODUCTION

Neural fuzzy network models can be used as global approximations in numerical optimization problems and therefore reduce the computational cost involved in expensive simulations. The use of grid partition implies in a high number of fuzzy rules when the number of input variables is augmented, thus, increasing the complexity of the model and the number of examples required for training. Clustered neural fuzzy models arise as an alternative to get round the so-called “curse of dimensionality”. Using clustering techniques it is possible to employ less fuzzy rules than with grid partition [1], [2]. However, it is difficult to choose the adequate number of clusters in a particular problem. Moreover, the cluster information may be used to refine the approximation at given regions of the input space.

This paper discusses the application of hierarchical clustering techniques in the design of neural fuzzy models for optimization. The paper

is organized as follows: first, we present the generation of clustered neural fuzzy approximations for optimization. Then, a number of hierarchical clustering methods is investigated for the function approximation problem, which is the main interest in optimization. Finally, an iterative design of the clustered neural fuzzy model is presented. Some optimization problems are analyzed for illustrating the proposed methodology.

CLUSTERED NEURAL FUZZY MODELS

Figure 1 shows schematically the application of clustered neural fuzzy models for generating a pseudo-objective function for the optimization step. The clustering techniques identify different clusters in the data, which form the fuzzy rules in a fuzzy inference system (FIS). The FIS generates a mathematical model for the data, which substitutes the real objective function in the optimization process.

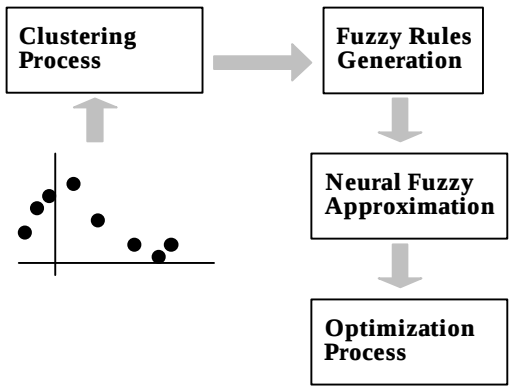

Figure 1. Generation of clustered neural fuzzy models for optimization.

The data set for the clustering process consists of the input vectors and their corresponding evaluations:

$$\Gamma = \{\mathbf{x}_t ; y_t\}_{t=1}^{T} \qquad (1)$$

Thus, each observation is a $(n+1)$-dimensional vector represented in the matrix:

$$\mathbf{X} = \begin{bmatrix} x_{11} & \cdots & x_{1n} & y_1 \\ \vdots & \ddots & \vdots & \vdots \\ x_{T1} & \cdots & x_{Tn} & y_T \end{bmatrix} \qquad (2)$$

The input data is, in general, uniformly distributed in the volume determined by the input space. By adding the information of function evaluations in the data set for clustering we are incorporating the information of the behavior of the function. The data is now distributed in a $(n+1)$-dimensional volume, but the function evaluations disperse the data accordingly to the function behavior. Thus, input points that are close to one another in the input space may not be in the $(n+1)$-dimensional space if the function has a rapid variation at these points. Figure 2 illustrates an example.

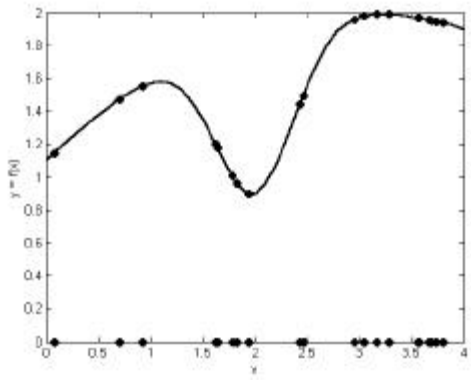

Figure 2. Example of a one-dimensional function. The data is dispersed along a curve in the two-dimensional space.

Observe that the input data, randomly distributed in the one-dimensional space may generate a given cluster distribution, which do not consider the information provided by the function evaluations. For example, the data for $1.5 < x < 2.0$ may be considered one cluster. However, adding the ordinate values, we have data distributed in a two-dimensional space, but the form of the distribution incorporates the information given by the function behavior. We can see in the graph that the data for $1.5 < x < 2.0$ are now distant because of the variation of the function in this region. The clusters formed in the $(n+1)$-dimensional space provide better information for the fuzzy rules generation.

After K clusters are identified, we calculate the centroids of each cluster, which can be

partitioned into the components of input and output dimensional space:

$$\mathbf{c}_i = [\bar{\mathbf{x}}_i \mid \bar{y}_i], \quad i = 1, \ldots, K \tag{3}$$

where $\bar{\mathbf{x}}_i$ are the components of the *n*-dimensional input space.

The *i*-th cluster centroid can then express the following fuzzy rule:

$$If\ (x_1\ is\ A_{i1}) and (x_2\ is\ A_{i2}) \cdots (x_n\ is\ A_{in})$$
$$then\ z_i\ is\ y\ (\mathbf{x})$$

where each input membership function is described by a Gaussian function:

$$A_{ij}(x_j) = \exp\left[-\frac{(x_j - \bar{x}_{ij})^2}{2s_{ij}^2}\right] \tag{4}$$

Thus, the degree to which a given vector $\mathbf{x}$ satisfies the *i*-th fuzzy rule is given by:

$$A_i(\mathbf{x}) = \prod_{j=1}^{n} A_{ij}(x_j) \tag{5}$$

$$A_i(\mathbf{x}) = \exp\left[-\sum_{j=1}^{n} \frac{(x_j - \bar{x}_{ij})^2}{2s_{ij}^2}\right] \tag{6}$$

The overall output of the model is:

$$z = \frac{\sum_{i=1}^{K} A_i(\mathbf{x}) z_i}{\sum_{k=1}^{K} A_k(\mathbf{x})} \tag{7}$$

where:

$$z_i = p_{i0} + \sum_{j=1}^{n} p_{ij} x_j \tag{8}$$

This model is equivalent to the Sugeno first order fuzzy inference system [1]. However, the clustered model comprises a fewer number of rules than the full grid partition model.

HIERARCHICAL CLUSTERING PROCESS

With the hierarchical methods for cluster analysis, the number of clusters is not determined *a priori*. This is a useful advantage in function approximation, since we do not know the behavior of the function that generated the data. However, the data for the function approximation problem is distributed along a hyper surface in a hyper volume. Thus, it is important to evaluate what is the most appropriate clustering technique for this type of data distribution, which is different from pattern classification problems.

In the hierarchical process, the observations are sequentially grouped by agglomerative or divisive procedures. The agglomerative procedures perform a sequence of fusions among the observations, whereas the divisive procedures perform successive divisions until obtaining refined clusters [6].

In this paper we consider the following agglomerative procedures:

1. Single linkage;
2. Complete linkage;
3. Average linkage;
4. Centroid method;
5. Ward method;

Next, we give a brief description of each method. For a detailed description, see [3]-[6].

Single Linkage

In all agglomerative procedures, the two clusters that present the least distance $d(C_p, C_q)$ in a given iteration are grouped into a new cluster. They differ in the definition of the distance $d(C_p, C_q)$.

In the Simple Linkage (SL) method, the distance between two clusters *p* and *q* is defined as the distance between the nearest observations:

$$d(C_p, C_q) = \min_{i \in C_p, j \in C_q} d(\mathbf{x}_i, \mathbf{x}_j) \tag{9}$$

Complete Linkage

In the Complete Linkage (CL) method, the distance between two clusters *p* and *q* is defined as the distance between the furthest observations:

$$d(C_p, C_q) = \max_{i \in C_p, j \in C_q} d(\mathbf{x}_i, \mathbf{x}_j) \tag{10}$$

Average Linkage

In the Average Linkage (AL) method, the distance is defined as the average distance between all pairs of observations:

$$d(C_p, C_q) = \frac{\sum_{i \in C_p} \sum_{j \in C_q} d(\mathbf{x}_i, \mathbf{x}_j)}{N_p N_q} \qquad (11)$$

where N_p is the number of observations in cluster C_p and N_q is the number of observations in C_q.

Centroid Method

In the Centroid Method (CM), the distance between two clusters is equal to the distance between their centroids:

$$d(C_p, C_q) = d(\bar{\mathbf{c}}_p, \bar{\mathbf{c}}_q) \qquad (12)$$

Ward Method

The Ward Method (WM) is similar to the Centroid Method, however the number of observations in each cluster weights the distance between the centroids:

$$d(C_p C_q) = \frac{N_p N_q}{N_p + N_q} d(\bar{\mathbf{c}}_p, \bar{\mathbf{c}}_q) \qquad (13)$$

DISTANCE METRICS

There are different metrics for measuring distances between two points $\mathbf{x}_i$ and $\mathbf{x}_j$. The traditional Euclidean distance, given by:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)'} \qquad (14)$$

The standardized Euclidean distance:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)\mathbf{S}^{-1}(\mathbf{x}_i - \mathbf{x}_j)'} \qquad (15)$$

where $\mathbf{S}$ is the diagonal matrix with diagonal elements given by the sample variances of the components of $\mathbf{x}$ over the T observations.

The City Block metric is given by:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right| \qquad (16)$$

Finally, the Mahalanobis metric is given by:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)\mathbf{V}^{-1}(\mathbf{x}_i - \mathbf{x}_j)'} \qquad (17)$$

where $\mathbf{V}$ is the sample covariance matrix.

ITERATIVE DESIGN OF CLUSTERED NEURAL FUZZY MODELS

Exploiting the clustering techniques we may formulate the following iterative scheme for improving the quality of the clustered neural fuzzy model:

1. Start with T_0 randomly distributed samples in the input space and then evaluate the function for each sample point, obtaining the initial data set Γ.
2. Then, the samples are normalized to the unitary hypercube before the clustering step.
3. Apply the clustering process.
4. Select the appropriate number of clusters.
5. Generate and train the neural fuzzy approximation.
6. If the quality of the approximation is acceptable then go to step 8, else go to step 7. The quality of the approximation is measured using a checking data set.
7. Generate more samples accordingly to a multidimensional Gaussian distribution centered at the centroid of the cluster with the highest checking error. Train the model again and go to step 6.
8. Optimize the neural fuzzy approximation.

The total number of samples used for generating the approximation is a compromise between quality of the approximation and computational cost. Thus, the user may also use a maximum number of samples allowable as a stop criterion. For deciding the appropriate number of clusters in step 4, one may use the distance $d(C_p, C_q)$ of the two clusters grouped in a given iteration (level of fusion). If a high increasing in the level of fusion occurs in a given

iteration, use the cluster division at the iteration immediately before [3].

Finally, observe that the maximum number of samples that the designer can accept limits the number of clusters in the model. This is because, in general, the least squares algorithm is used for tuning the linear parameters in (8). The neural fuzzy model has a total of $K(n+1)$ linear parameters. Thus, we need more than $K(n+1)$ samples for using the least squares estimator, since we need more equations than unknowns in the linear system. Therefore:

$$T > K(n+1) \Rightarrow K < T/(n+1) \qquad (18)$$

This criterion together with the level of fusion information may be employed to determine the number of clusters.

RESULTS

Analytical Problem I

We first investigate the approximation of the following nonlinear function:

$$f(\mathbf{x}) = 3(1-x_1)^2 \exp\left[-x_1^2 - (x_2+1)^2\right] - 10(0.2x_1 - x_1^3 - x_2^5)\exp\left[-x_1^2 - x_2^2\right] - (1/3)\exp\left[-(x_1+1)^2 - x_2^2\right] \qquad (19)$$

for $x_1, x_2 \in [-3,3]$. The function is shown in figure 3. The global minimum occurs in $\mathbf{x}^* = (0.2283, -1.6256)$ at which $f(\mathbf{x}^*) = -6.5511$.

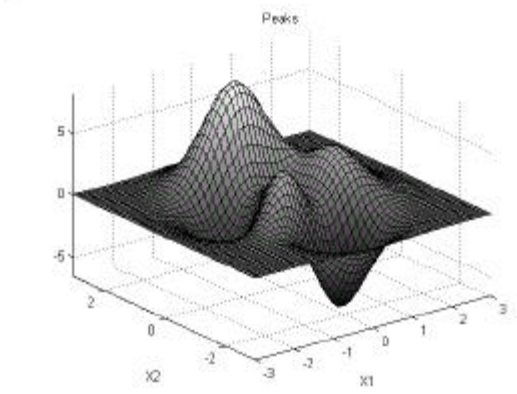

Figure 3. A nonlinear two-dimensional function.

We performed the following experiment:

1. First, we generate 100 randomly distributed samples in the input space and then evaluate the function for each sample point, obtaining the data set Γ.
2. Then, the samples are normalized to the unitary hypercube before the clustering step.
3. For each clustering technique and for four different distance metrics, we generate a neural fuzzy model for the data set Γ using eight clusters (thus, eight rules).
4. Finally, we measure the root mean square error (RMSE) before training, after 500 training epochs, and the RMSE over 21×21 points distributed in a uniform grid in the input space (check data).

We repeated the experiment 50 times, generating the data set again for each time.

Table 1 shows the mean values over the 50 runs of the experiment. The values for the initial training RMSE, the final training RMSE and the checking RMSE are shown. Each clustering technique is analyzed using four different metrics, respectively: the Euclidean distance, the standardized Euclidean distance, the City Block distance and the Mahalanobis distance.

Table 1. Results of the experiment

Method	Distance Metric	RMSE (1)	RMSE (500)	RMSE Check
SL	Euclid.	1.0524	0.3935	0.8088
	Std.Euclid.	1.1069	0.4610	1.0229
	City Block	1.0601	0.4314	1.1498
	Mahal.	1.1094	0.5241	1.2441
CL	Euclid.	1.0295	0.2957	0.7298
	Std.Euclid.	1.0347	0.3563	0.7892
	City Block	1.0283	0.2646	0.6616
	Mahal.	1.0423	0.3307	0.7579
AL	Euclid.	1.0220	0.2276	0.6457
	Std.Euclid.	1.0357	0.3402	0.6702
	City Block	1.0132	0.2825	0.7548
	Mahal.	1.0106	0.3457	0.8419
CM	Euclid.	1.0313	0.2855	0.7341
	Std.Euclid.	1.0516	0.3655	0.8480
	City Block	1.0127	0.2652	0.6992
	Mahal.	1.0161	0.3155	0.7197

WM	Euclid.	1.0150	0.2666	0.5969
	Std.Euclid.	1.0220	0.3727	0.8584
	City Block	0.9996	0.2506	0.5951
	Mahal.	1.0011	0.3246	0.8377

Observing the results in table 1 it is possible to see that the best overall combination is the Average Linkage with Euclidian distance. Regarding the checking RMSE, the best combinations are respectively Ward method with City Block distance, Ward method with Euclidian distance and the Average Linkage with Euclidian distance. It is important to notice that the good performance of the Euclidian distance is due to the normalization of the data to the unitary hypercube. This procedure equals the relative importance of each component of a given pattern. If this normalization is not used, the standardized Euclidian distance and the Mahalanobis distance will have better performances.

Minimizing the neural fuzzy approximations generated after the clustering step, we obtain the results shown in table 2, which may be compared to the optimal solution. The best optimization result was obtained from the model generated using the Ward method and the City Block distance.

Table 2. Results of the optimization

Method	**Distance Metric**	$\mathbf{X_1}$	$\mathbf{X_2}$	**FIS value**
AL	Euclid.	0.2470	-1.6669	-6.5941
WM	Euclid.	0.2163	-1.6700	-6.6720
WM	City Block	0.2264	-1.6250	-6.6227

This example has only two optimization variables. Therefore, the gain obtained in relation to the grid partition approach is small, since eight rules were employed.

Analytical Problem II

The next problem investigated consists in the minimization of the following function:

$$f(\mathbf{x}) = \sum_{i=1}^{n} 0.01\left[(x_i + 0.5)^4 - 30x_i^2 - 20x_i\right] \tag{20}$$

for $x_i \in [-6,6]$. Figure 4 depicts the function for $n = 2$. The global minimum is $x_i = -4.4538$, $i = 1,\ldots,n$. The function has 2^n local minima.

If we use three membership functions per input variable in the grid partition approach, we get a total of 3^n rules. This exponential characteristic is a strong drawback in neural fuzzy design. Thus, when increasing the dimension of the problem, the clustering approach becomes more practical.

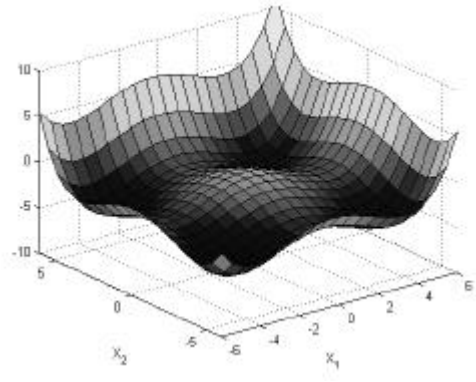

Figure 4. Function in (19) for $n = 2$.

Table 3 gives information about the clustered neural fuzzy approximations of (20) for $n = 2,\ldots,6$, where T is the number of sample points and K is the number of clusters. The clustering technique employed was the Ward method with Euclidian distance, which was one of the three best combinations in the previous problem, see table 1.

Table 4 shows the results of the optimization of the models in table 3. The real coded genetic algorithm was employed.

Table 3. Information about the clustered neural fuzzy models generated

n	**T**	**K**	**RMSE**	**RMSE (check)**
2	200	6	0.1274	0.2720
3	250	12	0.1921	0.3891
4	300	16	0.1886	0.2966
5	350	23	0.2430	0.3449
6	400	28	0.2690	0.4115

Table 4. Optimization results with respect to *n*

n	**X_1**	**X_2**	**X_3**	**X_4**	**X_5**	**X_6**
2	-4.363	-4.296	---	---	---	---
3	-4.465	-4.147	-4.250	---	---	---
4	-4.350	-4.101	-4.574	-4.837	---	---
5	-4.020	-4.377	-4.242	-3.312	-3.892	---
6	-3.210	-3.957	-4.196	-4.085	-4.842	-4.288

This example illustrates the advantage of using clustered neural fuzzy models for approximating functions in optimization problems. Furthermore, the solutions in table 4 may be improved starting a first order optimization method from the points achieved by the genetic algorithm.

Numerical Optimization Problem

Finally, we analyze the design of a superconducting magnetic energy storage – SMES – device with three design parameters. This problem consists in optimizing the dimensions of the external solenoid in a SMES with two solenoids. The objective is to minimize the strayed field evaluated in 21 points along the lines a and b, as illustrated in figure 5. The constraints are to maintain the stored energy in 180MJ and to satisfy the quench condition that guarantees the superconductivity state.

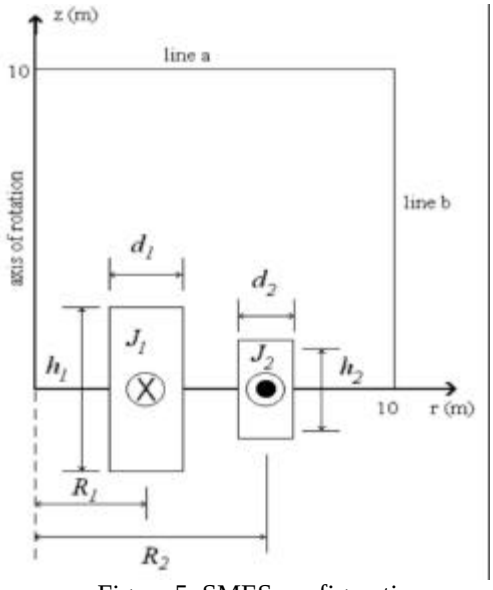

Figure 5. SMES configuration.

Mathematically, we have:

$$\min B_{stray} = \sqrt{\frac{1}{21}\sum_{i=1}^{21}\left|B_{stray,i}\right|^2} \qquad (21)$$

where $B_{stray,i}$ is the magnetic flux density calculated in the *i*-th point.

Also, the constraints are given by:

$$g_1(\mathbf{x}) = \frac{E}{E_0} - 1 = 0 \qquad (22)$$

$$g_2(\mathbf{x}) = B_{\max} - 4.92 \le 0 \qquad (23)$$

where the design parameters are $\mathbf{x} = [R_2 \quad h_2 \quad d_2]$, $E_0 = 180MJ$, and $B_{\max}$, in tesla, is the maximum magnetic flux density acceptable, otherwise the superconductivity state is violated. Table 5 summarizes information about the problem parameters.

Table 5. Parameters range and fixed parameters

	R1	**h1**	**d1**	**R2**	**h2**	**d2**
Units	*m*	*m*	*m*	*m*	*m*	*m*
Min	---	---	---	2.6	0.408	0.1
Max	---	---	---	3.4	2.2	0.4
Fixed	2.0	0.8	0.27	---	---	---

Also, the current densities are fixed and equal to:

$$\left|J_1\right| = \left|J_2\right| = 22.5MA/m^2 \qquad (23)$$

We deal with the constraints by defining the transformed objective function:

$$f(\mathbf{x}) = B_{stray} + 10g_1(\mathbf{x})^2 + 5\max\left[0, g_2(\mathbf{x})\right]^2 \qquad (24)$$

The evaluation of (21)-(23) involves the solution of an electromagnetic field problem, which is performed using the finite element method. An alternative approach for solving this problem is to generate a neural fuzzy approximation of (24) and optimize this

approximate model, which is less expensive than finite element analysis.

We first generate 300 randomly distributed samples and evaluate (24) for each sample point. After, the iterative design of the clustered neural fuzzy model is applied. We set the maximum number of samples equal to 400, i.e., more than 400 finite element analyses for generating the approximation is not permitted. In addition, using the relation in (18), we set the maximum number of clusters, and therefore rules, to 100. Figure 6 shows the plot of the levels of fusion during the clustering process.

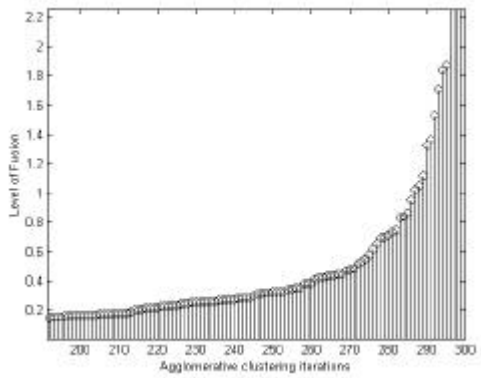

Figure 6. Level of fusion versus iterations.

It is possible to perceive "jumps" in the level of fusion after 283 iterations. In iteration 283 we have $300 - 283 = 17$ clusters. Thus, we cannot use less than 17 clusters. We have then a range of 17 to 100 for K. We select $K = 20$ clusters and generate a neural fuzzy model.

The training RMSE after 300 epochs was 0.1144 and the RMSE over a checking data with 80 samples was 0.0904. Table 6 reports the final optimization results using the real coded genetic algorithm coupled with the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method in a hybrid methodology. The total number of evaluations of the neural fuzzy approximation was 3037 (the last 37 evaluations are due to the BFGS method). Therefore, we obtain a rapid and efficient optimization process. We performed only 480 finite element analyses against the 3037 if the direct optimization would be used. Moreover, the number of linear parameters is 80 and the number of nonlinear parameters is 120, giving a total of 200 parameters. This number is to a great extent inferior to that in the grid partition approach.

Table 6. Final optimization results

	R2 [m]	**h2 [m]**	**d2 [m]**
Ref.[7]	3.0800	0.4780	0.3940
Value	2.9192	0.9739	0.2766
	Bstray [mT]	**Energy [MJ]**	**Bmax [T]**
Ref.[7]	0.8985	179.86	4.7341
Value	4.2	180.08	4.9014
	F	**G1**	**G2**
Ref.[7]	0.0009	–0.0008	–0.1859
Value	0.0042	0.0004	–0.0186

CONCLUSIONS

The use of hierarchical clustering techniques provide a rational manner for deciding the numbers of clusters, and therefore rules, needed for approximating an specific function. The iterative design process proposed in this paper exploits the cluster information using the error measure feedback to sample more points where the error is greater. This strategy permits to refine the model at given regions of the input space and to get round the drawbacks of the grid partition approach. The generation of accurate and inexpensive approximate models is a useful tool in practical optimization problems.

REFERENCES

1. J. S. R. Jang, C. T. Sun and E. Mizutani, *Neuro-fuzzy and Soft Computing*, Prentice Hall, New Jersey, 1997.

2. K. Rashid, J. A. Ramírez and E. M. Freeman, Optimization of Electromagnetic Devices Using Sensitivity Information from Clustered Neuro-Fuzzy Models, *IEEE Trans. Magn.*, **37**, 3575, (2001).

3. M. R. Anderberg, *Cluster Analysis for Applications*, Academic Press, New York, 1973.

4. B. S. Everitt, *Cluster Analysis*, Heineman Educational Books, London, 1993.

5. J. D. Jobson, *Applied Multivariate Data Analysis – Volume II: Categorical and Multivariate Methods*, Springer, New York, 1992.

6. S. Chiu, Fuzzy Model Identification Based on Cluster Estimation, *J. of Intelligent and Fuzzy Systems*, **2 (3)**, (1994).

7. TEAM Benchmark Problem 22 Definition, http://www.igte.tugraz.at/archive/team/index.htm.